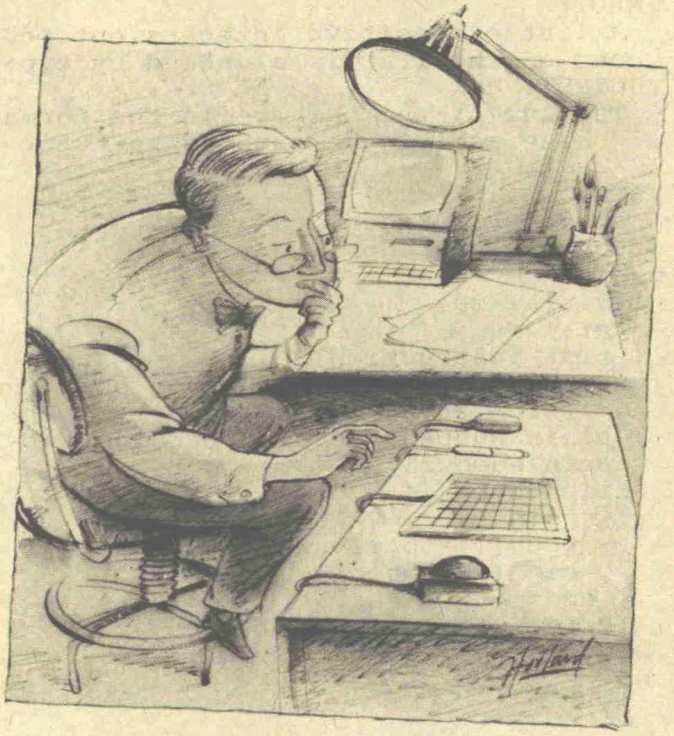
อาจหาญ สัตยารักษ์

SOFT MAIL

เทคนิคการนำไฟล์ข้อมูลไปไว้ในไฟล์ EXE



ปรับปรุงโปรแกรมเก่าที่เขียนในลักษณะอ่านข้อมูลข้างนอกมาเป็นการนำข้อมูลนั้น ๆ มาผนวกไว้ในไฟล์ EXE

มีความลับเรื่องหนึ่งที่ผู้อ่านหลายท่านอยากทราบเป็นอย่างมาก เพราะเมื่อใช้โปรแกรมสำเร็จที่พัฒนาโดยผู้เขียน เช่น THAI BANNER, THAISHOW ฯลฯ จะพบว่าไม่มีไฟล์ข้อมูลสำหรับแสดงอักขระภาษาไทยหรือภาษาอังกฤษปรากฏให้เห็นในแผ่นดิสก์ โดยเฉพาะใน THAI BANNER จะมีผู้สงสัยว่าภาพที่ใช้ในเมนูสี่ภาพนั้นมาจากไหน หลายท่านจึงมีจดหมายสอบถามไปยังผู้เขียน ซึ่งผู้เขียนก็ยังไม่เคยเปิดเผยเรื่องนี้ให้ใครทราบ

แต่สำหรับบทความนี้ ผู้อ่านจะได้ทราบความลับนั้น เพราะผู้เขียนขอนำเทคนิคที่ใช้นั้นมาเขียนให้ทราบ โดยแสดงให้เห็นขั้นตอนต่าง ๆ ซึ่งผู้เขียนยึดหลักที่ว่าเมื่อเราเป็นโปรแกรมเมอร์ การแก้ไขปัญหาต่าง ๆ นั้นจะทำแบบโปรแกรมเมอร์ทั้งหมด ไม่ใช่นำเอาระบบช่วยงานอื่น ๆ มาแก้ไข

ผู้เขียนได้จัดทำเป็นซอฟต์เมล์เช่นเดิม โดยจะมีไฟล์ดังนี้

1. ไฟล์ trandta1.pas, trandta2.pas, font1.csl, thai.pas เป็นการโอนไฟล์ข้อมูลในไฟล์ font1.csl ซึ่งเป็นฟอนต์อักขระบนจอภาพกราฟิกของโปรแกรมพิมพ์เอกสารราชวิถีเวิร์ดพีซี แสดงการโอนข้อมูล 2 แบบ และนำข้อมูลที่โอนได้จากแบบที่สองมาเขียนยูนิต thai.tpu (จากไฟล์ thai.pas)

2. ไฟล์ crut1.pic, crut2.pic, trancrut.pas, crut.pas ไฟล์เกี่ยวกับรูปครุฑเพื่อนำมาโอนข้อมูลเพื่อเรียกใช้ในไฟล์ crut.pas ทำให้พิมพ์รูปครุฑนี้ได้ด้วยเครื่องพิมพ์ 24 หัวเข็ม

3. ไฟล์ hercdrv.obj, vgadrv.obj ไฟล์ที่โอนมาจากไฟล์ *.bgi เดิม

หวังว่าการโอนทั้งสองแบบนี้จะช่วยให้ผู้อ่านเกิดความคิดที่จะนำไปโอนข้อมูลอื่น ๆ ได้ ทั้งนี้ขึ้นกับการให้ลำดับคำสั่งให้ตรงกับความต้องการ

การโอนข้อมูลจากไฟล์ font1.csl

ผู้เขียนได้เขียนไว้ในบทความเรื่อง "เมนูภาษาไทยกราฟิกกับการรันไฟล์" ซึ่งลงวารสารไมโครฯ ฉบับที่ 101 ในเรื่องการสร้างภาษาไทยกราฟิก โดยอาศัยข้อมูลจากไฟล์ของราชวิถีเวิร์ดพีซี font1.csl โดยได้อธิบายให้ทราบแล้วดังนี้

ข้อมูล 768 ไบต์แรกเป็นอักขระแสดง ctrl code

ข้อมูล 4728 ไบต์ถัดมาเป็นอักขระภาษาอังกฤษและภาษาไทยตามลำดับ (ผู้เขียนใช้เพียง 197 ชุด ๆ ละ 24 ไบต์) ซึ่งเป็นอักขระรหัสแอสกี 32-228 เท่านั้น เพราะผู้เขียนทำสระผสมด้วยเทคนิค

ของโปรแกรม

ผู้อ่านคงต้องไปหาวารสารฉบับที่ 101 มาเปิดดูโปรแกรมของบทความเรื่องนี้ใหม่อีกครั้ง

โปรแกรมในรูปที่ 1 คำสั่งบรรทัดที่ 7 คือ var font : array[0..4727] of byte; เป็นตัวแปรที่จะรองรับข้อมูลที่อ่านได้ ตามด้วยคำสั่งบรรทัดที่ 33-37 ซึ่งอ่านข้อมูลมาจากไฟล์ font1.csl ซึ่งหมายความว่าเราอ่านข้อมูลมาเก็บไว้ในตัวแปร font[0]-font[4727] นั่นเอง

เมื่อการอ่านทำงานดังนี้ เราย่อมพิจารณาได้ว่า

1. ถ้าเราตรวจค่าของข้อมูลใน font1.csl ได้ เราก็จะจดค่านั้น ๆ ได้ด้วย

2. เมื่อเราจดค่าที่ได้ในไฟล์ font-1.csl มาได้แล้ว เราย่อมเปลี่ยนค่าตัวแปรมิติ font ของเดิมให้กลับเป็นค่าที่ทราบแล้ว หรือที่เรารู้จักกันในค่าคงที่ (ในเทอร์โบปาสคาลเรียกว่า const)

3. เมื่อเราเปลี่ยนค่าที่อ่านให้มาเป็นค่าคงที่ได้ทำให้ไม่ต้องอ่านข้อมูลจากไฟล์นั้น ๆ อีกต่อไป

ตัวอย่างเช่น

1. อ่านข้อมูลมาใส่ตัวแปร

```
var pic:array[0..8] of byte;
```

โดยอ่านจากไฟล์ชื่อ pic.dta

2. ตรวจข้อมูลในไฟล์ pic.dta พบว่าเป็น 2, 7, 8, 27, 112, 0, 167, 11, 34 (โดยอาจใช้โปรแกรม pctools ตรวจสอบ)

3. ดังนั้นกำหนดใหม่ได้ดังนี้

```
const pic:array[0..8] of byte
= (2, 7, 8, 27, 112, 0, 167, 11, 34);
```

แต่ปัญหาที่สำคัญนั้นอยู่ที่ข้อมูลของไฟล์ font1.csl ที่เราต้องการนั้นมีถึง 4728 ไบต์ จึงเป็นไปไม่ได้ที่จะใช้ pctools ตรวจและจดข้อมูล 4728 ไบต์นี้ อีกประการหนึ่งคือ แม้จะพากเพียรจดมาได้ก็ยังจะต้องมาป้อนลงเป็นค่าคงที่อีก

มาถึงตรงนี้ผู้อ่านบางท่านคงนึกท้ออยู่ในใจไม่น้อย และอาจคิดเลยเถิดไปว่าผู้เขียนนี้ช่างขยันจังที่ทำได้ ซึ่งตามความเป็นจริงแล้ว ผู้เขียนมิได้ใช้แนวการทำงานด้วยการจดและการป้อนเองแม้แต่น้อย เพราะการทำอย่างนั้นโอกาสที่จะทำให้ข้อมูลผิดพลาดมีได้มากทีเดียว

ทางเลือกที่ผู้เขียนทำก็คือ การเขียนโปรแกรมช่วยในการโอนข้อมูลให้เป็นไปตามที่เราต้องการ โดยต้องคำนึงถึงสิ่งสำคัญต่าง ๆ ดังนี้

1. ข้อมูลที่โอนไปแล้วนั้นต้องสามารถเรียกมาดูบนเอดิเตอร์ของเทอร์โบปาสคาลได้ โดยหลักการที่เอดิเตอร์หรือโปรแกรมพิมพ์เอกสารต่าง ๆ นั้นมีรหัสจบบรรทัดปิดท้ายบรรทัดในจังหวะที่เหมาะสม รหัสจบบรรทัดนี้คือ รหัสแอสกี 13 ตามด้วยรหัสแอสกี 10 หรือ #13#10 ในเทอร์โบปาสคาล

2. ต้องใส่เครื่องหมายวงเล็บเปิด (และวงเล็บปิด) ในตำแหน่งที่ถูกต้องตามลักษณะค่าคงที่นั้น ๆ

3. ต้องแยกข้อมูลที่โอนนั้นออกเป็นส่วน ๆ ตามชนิดของข้อมูลที่กำหนด ซึ่งคือการใส่เครื่องหมาย , ขั้นข้อมูลนั้น ๆ

4. การทำงานตามแนวคิดของผู้เขียนนั้นมีจุดประสงค์ที่สำคัญอย่างหนึ่งคือ จะต้องโอนข้อมูลโดยเขียนเป็นข้อมูลที่สามารถเรียกใช้ในเทอร์โบปาสคาลด้วยคำสั่ง {$I ชื่อไฟล์} ได้ทันที โดยไม่ต้องมานั่งแก้ไขข้อมูลบนเอดิเตอร์ของปาสคาลอีก

ตามแนวคิดข้างต้นนี้จะเห็นว่าการทำงานนั้นจำเป็นต้องกำหนดชนิดของไฟล์ในการเขียนให้เป็นไฟล์ชนิดที่ 1 ระเบียนข้อมูลมี 1 เขตข้อมูล ๆ ละ 1 ไบต์ ซึ่งทำให้เราสามารถเขียนไฟล์นั้น ๆ ได้เหมือนกับการเขียนข้อความบนโปรแกรมพิมพ์เอกสารหรือบนเอดิเตอร์ของปาสคาลเอง

การโอนแบบที่หนึ่ง

เพื่อให้ผู้อ่านมองเห็นแนวการเขียนโปรแกรมโอนข้อมูลได้หลาย ๆ แบบ ผู้เขียนจะเขียนวิธีการโอนข้อมูลจากไฟล์ font1.csl ให้เห็นเป็น 2 แบบ โดยทั้งสองแบบนี้เมื่อโอนข้อมูลแล้วจะสามารถเรียกใช้ในปาสคาลได้ทันที (แต่รูปแบบการเรียกใช้จะต้องแตกต่างกัน)

การโอนแบบที่หนึ่งนี้ ผู้เขียนขอยึดโปรแกรมในบทความฉบับที่ 101 มาเป็นแนวทางในการโอน นั่นคือจะเป็นการเปลี่ยนข้อมูลจาก var font:array [0..4727] of byte ให้เปลี่ยนมาเป็นตัวแปรคงที่

เราต้องศึกษาคำสั่งเรียกไฟล์จากแผ่นดิสค์มาคอมพายล์ร่วม คือคำสั่ง {$I ชื่อไฟล์} นั้นว่ามีข้อจำกัดที่เราต้องระวังอย่างไรบ้าง ซึ่งจะมีข้อความคำนึงดังนี้

1. คำสั่งนี้จะเรียกไฟล์ที่มีความสมบูรณ์ในตัวไฟล์เองเท่านั้น คือ อาจเป็นชุดคำสั่ง ฟังก์ชัน ค่าคงที่ตัวแปรได้

2. ไฟล์ที่เรียกมานั้น จะเป็นส่วนหนึ่งของชุดคำสั่งไม่ได้ เช่น

```
Max:=9;
Total:=Max div 10;
```

ตามตัวอย่าง 2 บรรทัดข้างบนนี้ ถ้าใช้คำสั่ง {$I ชื่อไฟล์} เมื่อคอมพายล์โปรแกรมจะแจ้งข้อผิดพลาดว่า Error 118: Include files are not allowed here

3. ตำแหน่งการเรียกใช้คำสั่ง {$I ชื่อไฟล์} นั้นจะต้องอยู่ในตำแหน่งเดียวกันกับการเขียนโปรแกรมโดยปกติ

4. แต่ละบรรทัดในไฟล์จะต้องไม่ยาวจนเกินไป (คือต้องมีรหัสจบบรรทัดในตำแหน่งที่เหมาะสม) ซึ่งในเทอร์โบปาสคาลยอมให้ไม่เกิน 127 คอลัมน์ ถ้าเกินจะเกิด Error 11 : Line too long

การเขียนข้อมูลลงไฟล์นั้นเราต้องเข้าใจอีกเรื่องหนึ่งคือ ถ้าเราสั่งเขียนข้อมูลที่เป็นข้อความจะไม่สามารถสั่งเขียนลงไปทั้งข้อความได้ทีเดียว ทั้งนี้เพราะในเทอร์โบปาสคาลนั้นจะมีการเก็บความยาวข้อความในมิติ 0 ด้วย ดังนั้นการเขียนข้อความลงในไฟล์เพื่อให้สามารถใช้งานบนเอดิเตอร์ได้เลยจึงต้องใช้วิธีเขียนครั้งละหนึ่งไบต์ไล่จากต้นข้อความไปยังท้ายข้อความนั้น ๆ

โปรแกรมแรกที่จะศึกษาเป็นการโอนข้อมูลจากไฟล์ font1.csl ไปเป็นไฟล์ fontno1.pas ซึ่งจะเป็นค่าคงที่ const font:array[0..4727] of byte= โปรแกรมนี้คือไฟล์ trandta1.pas ดังรูปที่ 1 ขอให้ศึกษาแนวคิดและการทำงานของโปรแกรมด้วยตนเองก่อนอ่านคำอธิบาย

รายละเอียดในโปรแกรมรูปที่ 1 มีดังนี้

1. บรรทัดที่ 3 กำหนดตัวแปรเป็นมิติ 1-4728 ทั้งนี้เพื่อสะดวกต่อการกำหนดรหัสจบบรรทัด

2. บรรทัดที่ 4-12 อ่านข้อมูลจากไฟล์ font1.csl มาใส่ในตัวแปร font ในขนาด 4728 ไบต์

```
 1: PROGRAM TranDta1; {==Trandata from font1.csl to fontno1.pas ==}
 2: uses crt,dos;
 3: var  font:array[1..4728] of byte;

 4: PROCEDURE Readfont;
 5: var f:file;
 6: begin
 7:   assign(f,'font1.csl');
 8:   reset(f,1);
 9:   blockread(f,font,768);
10:   blockread(f,font,4728);
11:   close (f);
12: end;

13: PROCEDURE TranData;
14: var NO:word;   f:file;
15:     ST:string;
16:        Procedure SaveData(STT:string);
17:        var kk:byte;
18:        begin
19:           for kk:=1 to length(STT) do
20:                 blockwrite(f,STT[kk],1);
21:        end;
22: begin
23:   assign(f,'Fontno1.pas');
24:   rewrite(f); reset(f,1);
25:   Savedata('Const  font:array[0..4727] of byte='#13#10+'   (');
26:   for No:=1 to 4728 do
27:     begin
28:        writeln(No);
29:        str(Font[No]:3,ST);
30:        if ST[1]=#32 then ST[1]:='0';
31:        if ST[2]=#32 then ST[2]:='0';
32:        if No<4728 then ST:=ST+','
33:                   else ST:=ST+');'#13#10#26;
34:        SaveData(ST);
35:        if No mod 10=0 then SaveData(#13#10+'    ');
36:     end;
37: end;

38: {=================== TRANDTA1.PAS ====================}
39: begin
40:  Readfont;
41:  TranData;
42: end.
```

รูปที่ 1 การเปลี่ยนข้อมูลจากไฟล์ font1.csl ให้เป็นข้อมูลในภาษาปาสคาล

3. บรรทัดที่ 13-37 ส่วนการเปลี่ยนข้อมูลจากตัวแปร font ให้เป็นค่าคงที่ และเขียนลงในไฟล์ใหม่ในชื่อ fontno1.pas

4. บรรทัดที่ 16-21 ส่วนการเขียนข้อมูลที่คำนวณได้ลงในไฟล์ fontno1.pas ซึ่งจะเห็นว่าเป็นการเขียนค่าในตัวแปร STT ลงครั้งละหนึ่งไบต์ตามคำสั่ง blockwrite(f, STT[kk], 1); ซึ่งจะทำให้ไม่เขียนส่วนของความยาวข้อความ (STT[0]) ลงไปด้วย

5. บรรทัดที่ 23-24 กำหนดไฟล์

fontno1.pas โดยเป็นการเขียนไฟล์ใหม่ พร้อมกับเปิดไฟล์เพื่อรอการเขียนข้อมูลลงไฟล์

6. บรรทัดที่ 25 เขียนข้อความบรรทัดแรกและส่วนต้นของบรรทัดที่สองลงในไฟล์ fontno1.pas ขอให้สังเกต #13#10 ที่แทรกอยู่ในข้อความ ซึ่งหมายถึงการกำหนดให้ขึ้นบรรทัดใหม่ที่หลังรหัสสองตัวนี้ ดังนั้นในขณะนี้ไฟล์ fontno1.pas พร้อมจะเขียนข้อมูลต่อในบรรทัดที่ 2 โดยต่อจากเครื่องหมาย (ที่เขียนไว้นี้

7. บรรทัดที่ 26-36 ส่วนการโอนตัวแปร font ไปเป็นค่าคงที่ในไฟล์ fontno1.pas ดังนี้

- บรรทัดที่ 29-31 เปลี่ยนตัวแปร font[no] ให้เป็นข้อความ ST โดยกำหนดให้มี 3 ไบต์ (ไม่นับความยาวของข้อความ) พร้อมกับตรวจสอบ ST[1], ST[2] เพื่อเปลี่ยนค่าให้เป็นเลข 0

- บรรทัดที่ 32-33 กำหนดให้เครื่องหมาย , ต่อท้ายตัวแปร ST ยกเว้นกรณีที่เป็น font[4728] เท่านั้นที่กำหนดให้ต่อท้ายตัวแปร ST ด้วย ');'#13#10#26; ทั้งนี้เพราะว่าโดยลักษณะของค่าคงที่ที่เราได้กำหนดไว้นี้จะมีเครื่องหมาย , ขั้นระหว่างข้อมูลติดกัน ยกเว้นข้อมูลสุดท้ายที่จะปิดด้วย); ส่วนรหัส #26 นั้นคือรหัสปิดท้ายไฟล์ หรือที่เรารู้จักกันดีว่าคือรหัส EOF

- บรรทัดที่ 34 เขียนข้อมูลจากตัวแปร ST ที่คำนวณได้ลงไฟล์ fontno1.pas

- บรรทัดที่ 35 ทุกข้อมูลลำดับที่ 10 ให้เลื่อนบรรทัดใหม่เสมอ ช่องว่างที่ตามหลัง #10 นั้นเพื่อให้ตำแหน่งของข้อมูลไฟล์ fontno1.pas มีความสวยงามและตรวจสอบได้ง่าย

8. บรรทัดที่ 39-42 ส่วนโปรแกรมหลักเพื่อทำงานโอนข้อมูล

```
Const  font:array[0..4727] of byte=
    (000,000,000,000,000,000,000,000,000,000,
     000,000,000,000,000,000,000,000,000,000,
     000,000,000,000,000,000,000,000,000,000,
     000,024,060,060,060,024,024,000,024,024,
                    .
                    .
     000,000,000,000,000,000,000,003,012,020,
     008,000,000,000,000,000,000,000,000,000,
     000,000,000,000,000,000,000,000);
```

รูปที่ 2 แสดงส่วนหนึ่งของข้อมูลในไฟล์ fontno1.pas ที่โอนมาได้

มาถึงตรงนี้ผู้อ่านคงจะเกิดความคิดอีกมากมาย บางท่านอาจนึกโกรธตัวเองที่เคยหลงทำงานบางอย่างด้วยการบันทึกด้วยปากกามาหลายครั้ง ทั้ง ๆ ที่ความจริงเราได้เปรียบกว่าผู้ใช้รายอื่น ๆ ที่เขียนโปรแกรมได้ด้วยโปรแกรมสั้น ๆ เพียง 42 บรรทัดแต่สามารถทำงานแทนเราได้อย่างมีประสิทธิภาพ สะดวก และได้ข้อมูลที่ถูกต้อง

ดังนั้นคราวต่อไปก่อนจะรันไฟล์ pctools เพื่อเลือกไฟล์แล้วจับปากกามาจดข้อมูลที่อ่านบนจอภาพ แล้วมาป้อนข้อมูลกันต่ออีกที ก็ขอให้ลองนึกถึงบทความนี้ก่อนนะครับ

เมื่อรันไฟล์ trandta1.pas เราจะได้ไฟล์ fontno1.pas ในแผ่นดิสก์ซึ่งมีความยาว 21789 ไบต์ ซึ่งมีลักษณะของข้อมูลดังรูปที่ 2 (ตัดมาเพียงส่วนหนึ่งเท่านั้น)

การใช้งานไฟล์ fontno1.pas

ดังที่กล่าวมาแต่ต้นแล้วว่าไฟล์ fontno1.pas นี้เราสร้างเพื่อใช้ในไฟล์ mainmenu.pas ของบทความเรื่อง "เมนูภาษาไทยกราฟิกกับการรันไฟล์" ในวารสารไมโครฯฉบับที่ 101 เราจึงสามารถปรับปรุงให้ทำงานใหม่ได้ดังนี้

1. เพิ่มคำสั่งโดยให้แทรกก่อนบรรทัดที่ 3 ดังนี้

```
{$I fontno1}
```

2. ตัดบรรทัดที่ 7 ออกไป เพราะว่าเราได้กำหนดตัวแปร font ให้เป็นค่าคงที่แล้ว (ถ้าไม่ตัดจะซ้ำกัน)

3. ตัดบรรทัดที่ 33-37 ออกไป คือยกเลิกการอ่านข้อมูลจากไฟล์ font1.csl เพราะเรานำข้อมูลที่ต้องการนั้นมากำหนดเป็นค่าคงที่ในไฟล์ fontno1.pas แล้ว

เมื่อคอมพายล์จะได้ไฟล์ mainmenu.exe ที่ดีกว่าในบทความฉบับดังกล่าวมาใช้งาน เพราะว่าไฟล์ที่ได้นี้ไม่ต้องใช้ไฟล์ *.bgi และ font1.csl อีกต่อไป

การโอนแบบที่สอง

เพื่อให้ผู้อ่านเกิดความคิดแตกฉานมากขึ้น ผู้เขียนจึงขอเสนอแนวทางในการโอนจากไฟล์ font1.csl อีกแบบหนึ่งดังนี้

1. ไฟล์ข้อมูลที่อ่านได้มีขนาด 4728 ไบต์ โดยเป็นรหัสแอสกี 32-228 จึงมีอยู่ 228 − 32+1=197 ชุด

2. เราทราบตามบทความฉบับที่ 101 แล้วว่าข้อมูลหนึ่งอักขระนั้นมี 24 ไบต์

3. ดังนั้นแทนที่เราจะกำหนดแบบแรกแล้ว เรายังสามารถกำหนดเป็นค่าคงที่ชนิด 2 มิติได้คือ

```
const font:array[32..228, 1..24] of byte=
```

4. เพื่อไม่ให้ข้อมูลที่โอนได้ซ้ำกับวิธีแรก เราจะโอนไปยังไฟล์ชื่อ fontno2.pas แทน

5. เมื่อเป็นการโอนข้อมูลเป็นชนิด 2 มิติ เราก็สามารถทำได้โดยการวนรอบ for สองชั้น ซึ่งจะทำให้เรารู้ว่าเมื่อใดเริ่มข้อมูลและสิ้นสุดข้อมูลแต่ละชุดได้

เราต้องพิจารณารูปแบบของค่าคงที่ 2 มิติก่อน ซึ่งจะกำหนดได้ดังตัวอย่าง

```
const font:array[1..3, 1..6] of byte =
((112, 034, 123, 011, 022, 024), (242, 078,
211, 073, 093, 067), (082, 052, 150, 096,
068, 094));
```

ข้อมูลแต่ละชุดมีเครื่องหมาย , สำหรับแยกข้อมูลออกจากกัน และมีเครื่องหมาย ( ) ปิดหัวท้ายของข้อมูลแต่ละชุด และใช้เครื่องหมาย , แยกข้อมูลแต่ละชุดออกจากกันอีกด้วย

ดังนั้นการเขียนโปรแกรมให้โอนข้อมูลในแบบที่สองนี้จะต้องสามารถทำได้ตามรูปแบบของค่าคงที่ 2 มิติ ซึ่งเมื่อเขียนโปรแกรมจะได้ดังไฟล์ trandta2.pas ในรูปที่ 3 มีรายละเอียดดังนี้

1. บรรทัดที่ 3 กำหนดข้อมูลให้

```
 1: PROGRAM TranDta2; {== TranData from font1.csl to fontno2.pas ==}
 2: uses crt,dos;
 3: var  font:array[32..228,1..24] of byte;

 4: PROCEDURE Readfont;
 5: var f:file;
 6: begin
 7:   assign(f,'font1.csl');
 8:   reset(f,1);
 9:   blockread(f,font,768);
10:   blockread(f,font,4728);
11:   close (f);
12: end;

13: PROCEDURE TranData;
14: var No,k:byte;   f:file;
15:     ST:string;
16:       Procedure SaveData(STT:string);
17:       var kk:byte;
18:       begin
19:          for kk:=1 to length(STT) do
20:                blockwrite(f,STT[kk],1);
21:       end;
22: begin
23:   assign(f,'Fontno2.pas');
24:   rewrite(f); reset(f,1);
25:   Savedata('Const  font:array[32..228,1..24] of byte='#13#10+'   (');
26:   for No:=32 to 228 do
27:     begin
28:       write(No,'==>');
29:       for k:=1 to 24 do
30:        begin
31:          write(' ',k);
32:          str(Font[No][k]:3,ST);
33:          if ST[1]=#32 then ST[1]:='0';
34:          if ST[2]=#32 then ST[2]:='0';
35:          case k of
36:            1:ST:='('+ST+',';
37:            12:ST:=ST+','#13#10+'     ';
38:            2..23:ST:=ST+',';
39:            24:if No=228 then ST:=ST+'));'#13#10#26
40:                         else ST:=ST+'),'#13#10+'    ';
41:          end;
42:          SaveData(ST);
43:        end;
44:        writeln;
45:     end;
46: end:
47: {=================== TRANDTA2.PAS ====================}
48: begin
49:  clrscr;
50:  Readfont;
51:  TranData;
52: end.
```

รูปที่ 3 การเปลี่ยนข้อมูลจากไฟล์ font1.csl อีกแบบหนึ่ง

```
Const  font:array[32..228,1..24] of byte=
    ((000,000,000,000,000,000,000,000,000,000,000,000,
      000,000,000,000,000,000,000,000,000,000,000,000),
     (000,000,000,000,000,000,000,024,060,060,060,024,
      024,000,024,024,000,000,000,000,000,000,000,000),
                          .
                          .
     (000,000,000,003,012,020,008,000,000,000,000,000,
      000,000,000,000,000,000,000,000,000,000,000,000));
```

รูปที่ 4 แสดงส่วนหนึ่งของข้อมูลในไฟล์ fontno2.pas ที่โอนมาได้

เป็นตัวแปร 2 มิติด้วย

2. บรรทัดที่ 4-12 อ่านข้อมูลเข้ามาในตัวแปร font ที่กำหนด ซึ่งยังคงเหมือนเดิม

3. บรรทัดที่ 13-46 ส่วนการโอนข้อมูล

4. บรรทัดที่ 16-21 การเขียนข้อมูลลงในไฟล์ fontno2.pas

5. บรรทัดที่ 23-24 เปิดไฟล์ fontno2.pas

6. บรรทัดที่ 25 เขียนข้อมูลบรรทัดแรกและบรรทัดที่สองของไฟล์ fontno2.pas ให้เป็นค่าคงที่แบบ 2 มิติ

7. บรรทัดที่ 26-45 วนรอบโอนข้อมูลไปเก็บในไฟล์ fontno2.pas โดยตรวจสอบข้อมูลแต่ละลำดับ ซึ่งจะเก็บแตกต่างกัน

8. บรรทัดที่ 36 ข้อมูลลำดับแรกจะต้องใส่เครื่องหมาย (ข้างหน้า และใส่, ข้างหลัง)

9. บรรทัดที่ 37 ข้อมูลลำดับที่ 12 จะเลื่อนบรรทัดพร้อมกับให้ช่องว่างเริ่มต้นที่บรรทัดถัดไป

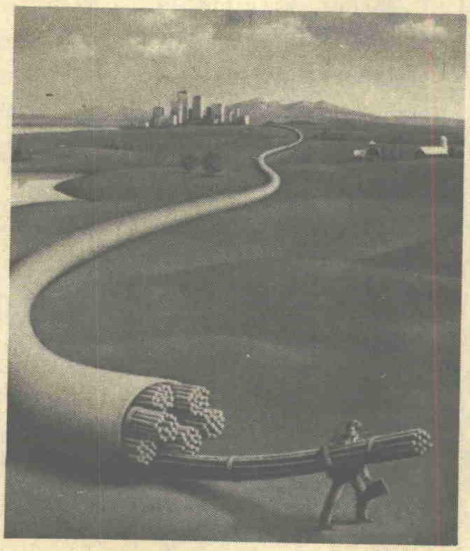
10. บรรทัดที่ 38 ข้อมูลลำดับที่ 2-23 (ยกเว้น 12 ซึ่งถูกทำไปแล้วจากบรรทัดที่ 37) จะเพิ่ม , ที่ข้างหลังตัวแปร ST เพียงอย่างเดียว

(การที่ข้อมูลที่ 12 ไม่ถูกกระทำด้วยบรรทัดที่ 38 เพราะเป็นการทำงานตามเงื่อนไขของ case)

11. บรรทัดที่ 39-40 ข้อมูลลำดับที่ 24 จะทำงาน 2 แบบ ขึ้นกับว่าเป็นส่วนปิดท้ายหรือไม่

12. บรรทัดที่ 48-52 ส่วนโปรแกรมหลักเพื่อโอนข้อมูล

เมื่อรันไฟล์ trandta2.pas เราจะได้ไฟล์ fontno2.pas ในแผ่นดิสก์ซึ่งมีความยาว 21911 ไบต์ ซึ่งมีลักษณะของข้อมูลดังรูปที่ 4 (ตัดมาเพียงส่วนหนึ่งเท่านั้น)



นำข้อมูลมาสร้างยูนิต THAI.TPU

เพื่อให้เห็นตัวอย่างการใช้งานไฟล์ fontno2.pas ที่โอนมาได้ ผู้เขียนขอเสนอยูนิต thai.tpu ซึ่งได้ดัดแปลงบางส่วนจากบทความในฉบับที่ 101 ให้มาใช้ค่าคงที่ font แบบ 2 มิติที่อยู่ในไฟล์ fontno2.pas โดยเป็นยูนิตง่าย ๆ (เพราะไม่ต้องการให้โปรแกรมยาวเกินความจำเป็น) ดังนั้นผู้อ่านต้องปรับปรุงโปรแกรมให้มีความสมบูรณ์ตามความต้องการเอง

การทำงานที่ต้องเปลี่ยนแปลงนั้นมีดังนี้

1. การสลับข้อมูลในกรณีที่รันกับจอสีหรือจอวีจีเอโมโนโครม

2. การแสดงอักขระกราฟิกบนจอภาพจากตัวแปร 2 มิติ ซึ่งจะต้องกำหนดให้แตกต่างจากข้อมูลที่เป็นมิติเดียว

เนื่องจากการกำหนดค่าคงที่นั้นได้กำหนดตามรหัสแอสกีอยู่แล้ว ดังนั้นจึงได้

font[32] คือรหัสแอสกี 32

font[49] คือเลข 1

font[65] คือ A

font[161] คือ ก

หมายความว่าเราหาข้อมูลของอักขระนั้นได้จาก font[ASCII] ถ้า ASCII คือรหัสแอสกีของอักขระที่ต้องการแสดงผลบนจอภาพ

นำมาเขียนโปรแกรมได้ดังรูปที่ 5 ซึ่งผู้เขียนขอจะไม่อธิบายการทำงานของ

โปรแกรม จึงขอให้ศึกษาด้วยตนเอง ไฟล์ thai.tpu ที่ได้จากการคอมพายล์ไฟล์นี้ ผู้เขียนจะนำไปใช้ในการโอนข้อมูลรูปภาพต่อไป ในกรณีที่ต้องการใช้กราฟิกขนาด 640×480 จุดให้ตัด Grmode:=1; ในบรรทัดที่ 28 ออกไป การที่ผู้เขียนได้กำหนดให้ GrDriver:=9; เมื่อตรวจไม่พบว่าเป็นจอโมโนโครมนั้นเกิดจากสาเหตุที่ในปัจจุบันนี้มีการนำการ์ด EGA มาใช้กับจอจีวีเอโมโนโครม (โดยผู้ซื้อไม่ทราบ) จึงได้กำหนดให้เปลี่ยนการ์ดนั้นให้แสดงผลเป็นการ์ด VGA เสียก่อน

โอนข้อมูลจากภาพหน้าจอ

คราวนี้มาถึงเรื่องการโอนข้อมูลจากจอภาพกันบ้าง เหตุที่ผู้เขียนเลือกการโอนข้อมูลจากจอภาพมาเป็นตัวอย่างนั้นเพราะว่า ผู้อ่านอาจใช้ในการโอนข้อมูลภาพได้ด้วย เพียงแต่ต้องหาทางอ่านภาพนั้น ๆ มาแสดงบนจอภาพให้ได้เสียก่อน เมื่อได้ภาพปรากฏบนจอแล้วก็ให้ใช้แนวคิดนี้ในการโอนข้อมูลไปเก็บไว้ได้

ในตัวอย่างที่จะกล่าวถึงต่อไปนี้ ผู้เขียนใช้ภาพตัวอย่างที่ให้มาคือ crut1.pic, crut2.pic ซึ่งเป็นภาพที่ผู้เขียนสแกนภาพครุฑเข้ามา 2 ขนาด แต่ภาพที่ให้มานี้เป็นภาพที่ผู้เขียนได้เก็บจากจอภาพมาง่าย ๆ โดยใช้คำสั่ง getimage เก็บภาพแล้วเขียนลงไฟล์อีกที ดังนั้นการเรียกให้มาแสดงบนจอภาพจึงเป็นการใช้คำสั่ง putimage มาแสดงบนจอภาพอีกที

หากผู้อ่านต้องการโอนภาพอื่น ๆ ก็ให้หาทางนำภาพนั้น ๆ ขึ้นมาบนจอภาพให้ได้ก่อนเท่านั้น

การโอนข้อมูลในกรณีที่ผู้เขียน

รูปที่ 5 การนำข้อมูลจากไฟล์ fontno2.pas มาสร้างระบบภาษาไทย

```
 1: UNIT Thai;
 2: INTERFACE
 3: uses crt,dos,graph;
 4: var MonoScn:boolean;   Regs:registers;
 5:     Ch,Funckey:char;
 6: PROCEDURE Outthaixy(column,row:integer; content:string);
 7: PROCEDURE ClsGR(x1,y1,x2,y2:integer);
 8: PROCEDURE ReadChar;
 9: IMPLEMENTATION
10: {$I fontno2}

11: FUNCTION SwapBit(B:Byte):byte;
12: begin
13:  SwapBit:=B and 1 shl 7 + B and 2 shl 5  + B and 4 shl 3  +
14:           B and 8 shl 1 + B and 16 shr 1 + B and 32 shr 3 +
15:           B and 64 shr 5 + B and 128 shr 7;
16: end;

17: PROCEDURE HercDriver; external;
18:   {$L Hercdrv.obj}
19: PROCEDURE EgaVgaDriver; external;
20:   {$L Vgadrv.obj}
21: PROCEDURE OpenGraph;
22: var GrDriver,Graphck,Grmode,no,k:integer;
23: begin
24:   MonoSCN:=False;  DetectGraph(GrDriver,GrMode);
25:   if GrDriver=7 then MonoSCN:=true
26:    else
27:     begin
28:      GrDriver:=9; Grmode:=1;
29:     end;
30:   if MonoSCN then Graphck:=RegisterBgidriver(@HercDriver)
31:              else Graphck:=RegisterBgidriver(@EgavgaDriver);
32:   InitGraph(GrDriver, GrMode, '');
33:   if pred(Monoscn) then
34:     for no:=32 to 228 do
35:       for k:=1 to 24 do font[no][k]:=swapbit(Font[no][k]);
36: end;

37: PROCEDURE Showchar(x,y:integer; ASCII:byte);
38: var k:integer;
39: begin
40:  for k:=1 to 24 do
41:   begin
42:    if font[ASCII][k]<>0 then
43:     begin
44:       if monoscn then setlinestyle(4,font[ASCII][k] shl 7,1)
45:                  else setlinestyle(4,font[ASCII][k],1);
46:       Line(x,k+y,x+15,k+y);
47:     end;
48:   end;
49: end;

50: PROCEDURE outthaixy(column,row:integer; content:string);
51: var k,ypix,xpix:integer; color:byte;
52: begin
53:  if Content[1]=#17 then delete(Content,1,1)
54:   else
55:    begin
```

รูปที่ 5 (ต่อ)

```
56:       column:=(column-1)*9;  row:=(row-1)*14;
57:     end;
58:    for k:=1 to length(content) do
59:     begin
60:       ypix:=0; xpix:=0;
61:       if content[k]=#3 then
62:         begin  inc(k);
63:          if content[k] in ['0'..'9'] then
64:           begin
65:            color:=ord(content[k])-48; inc(k);
66:            if content[k]=#1 then
67:                begin inc(color,9); inc(k); end;
68:            setcolor(color);
69:           end;
70:         end;
71:       if content[k]=#207 then
72:         begin
73:           showchar(column-9,row,222); content[k]:=#206;
74:         end;
75:       if content[k] in [#217..#228] then xpix:=-9;
76:       if content[k] in [#215,#216]  then xpix:=-8;
77:       if ((content[k])>#223) and ((content[k-1]>#216) or
78:                   (content[k+1]=#207))  then ypix:=-3;
79:       if content[k] in [#32..#228] then
80:           showchar(column+xpix,row+ypix,ord(content[k]));
81:       if content[k] in [#32..#214] then inc(column,9);
82:     end;
83:      setlinestyle(0,255,1);
84: end;

85: PROCEDURE ClsGR(x1,y1,x2,y2:integer);
86: var ViewPort:ViewPortType;
87: begin
88:    GetViewSettings(ViewPort);
89:    setviewport(x1,y1,x2,y2,clipon);  clearviewport;
90:    setviewport(ViewPort.x1,ViewPort.y1,ViewPort.x2,
91:                ViewPort.y2,ViewPort.Clip);
92: end;
93: PROCEDURE ReadChar;
94: begin
95:   Regs.AH:=$0;  intr($16,regs);
96:   Funckey:=#0;  ch:=chr(regs.AL);
97:   if ch=#0 then Funckey:=chr(regs.AH);
98: end;

99: {================= THAI.PAS ====================}
100: begin
101:  Opengraph;
102: end.
```

จะแสดงเป็นตัวอย่างนี้ เป็นการโอนข้อมูลเก็บในไฟล์ crutdata.pas เพื่อใช้นำไปพิมพ์ด้วยเครื่องพิมพ์ 24 หัวเข็ม จึงต้องมีหลักการในการโอนภาพดังนี้

1. เลือกกรอบภาพที่จะโอนข้อมูล โดยผู้อ่านอาจใช้คำสั่ง rectangle เขียนสี่เหลี่ยมโปร่งดูขอบเขตของภาพที่ต้องการโอนเสียก่อน

2. ให้ค่าของขอบเขตมุมบนซ้าย ดังนี้ Xleft เป็นค่าแกน x และ Yup เป็นค่าในแกน y

3. ให้ค่าของขอบเขตมุมล่างขวา ดังนี้ Xright เป็นค่าแกน x และ Ydown เป็นค่าในแกน y

4. คำนวณหาจำนวนครั้งในการพิมพ์ โดยใช้หลักการพิมพ์กราฟิก โดยที่ข้อมูล 24 บิตนับจากบนลงล่างจะตรงกับหัวเข็มทั้ง 24 หัวเข็ม โดยหาได้จาก Ydown-Yup+1 หารด้วย 24

5. การเก็บข้อมูลจะเก็บครั้งละ 3 ไบต์ในแนวดิ่ง โดยหนึ่งไบต์จะแทนจุด 8 จุด โดยจะเก็บอย่างนี้ไปเรื่อย ๆ จาก Xleft ไปจนถึง Xright

6. เมื่อเก็บแต่ละ 24 แถวเรียงจากซ้ายไปขวาจนครบแล้ว ก็จะขยับมาเก็บข้อมูลในแถวที่ต่ำจากกลุ่มที่เก็บไปแล้ว 24 แถวเรื่อย ๆ ไปจนครบตามขอบเขตที่กำหนด

นำแนวคิดข้างต้นมาเขียนโปรแกรมโอนข้อมูลจากจอภาพเพื่อไปพิมพ์ลงกระดาษได้ดังโปรแกรมในรูปที่ 6 (ไฟล์ในซอฟต์เมล์ชื่อ trancrut.pas) มีรายละเอียดดังนี้

1. บรรทัดที่ 4 กำหนดตัวแปร pic ขนาด 22577 ไบต์เพื่อรองรับภาพ ครุฑ (จากไฟล์ crut1.pic)

2. บรรทัดที่ 5 กำหนดตัวแปรที่ใช้งาน

3. บรรทัดที่ 6-13 อ่านข้อมูลภาพครุฑที่เก็บด้วยคำสั่ง getimage มาในหน่วยความจำแล้วแสดงภาพด้วยคำสั่ง putimage ซึ่งผู้เขียนกำหนดให้แสดงที่มุมบนซ้ายของจอภาพ จึงทำให้มุมบนซ้ายอยู่ที่จุด 0, 0 ซึ่งมีผลให้ตัวแปร Xleft=0, Yup=0

โดยการเขียนคำสั่ง rectangle - พบว่าเมื่อใช้ rectangle (0, 0, 200, 215); จะได้สี่เหลี่ยมโปร่งครอบคลุมรูปครุฑพอดี โปรดสังเกตว่า 215−0+1=216

รูปที่ 6
การเปลี่ยนข้อมูลภาพจากจอภาพไปเป็นไฟล์ข้อมูลสำหรับพิมพ์

```
 1: PROGRAM TranCrutDATA;
 2: uses crt,thai,graph,dos;
 3: var f:file;
 4:     pic:array[0..22576] of byte;
 5:     Xwidth,Ywidth,Xleft,Xright,Yup,Ydown:word;

 6: PROCEDURE ReadCrut;
 7: begin
 8:   assign(f,'crut1.pic');
 9:   reset(f,1);
10:   blockread(f,pic,filesize(f));
11:   close(f);
12:   putimage(0,0,pic,0);
13: end;

14: PROCEDURE TranData;
15: const BitK:array[0..7] of byte=(128,64,32,16,8,4,2,1);
16: var x,y,No,k:word;  ST:string;
17:     CrutData,Round:byte;
18:      Procedure SaveData(STT:string);
19:        var kk:byte;
20:        begin
21:           for kk:=1 to length(STT) do
22:                 blockwrite(f,STT[kk],1);
23:        end;
24: begin
25:    assign(f,'crutdata.pas');
26:    rewrite(f); reset(f,1);
27:    Xleft:=0;       Yup:=0;
28:    Xright:=200;    Ydown:=215;
29:    if (Ydown-Yup+1) mod 24<>0 then
30:          Ydown:=Ydown+23-Ydown mod 24+yup;
31:    Ywidth:=(Ydown-Yup+1) div 24;
32:    Str(Ywidth,ST); SaveData('Const Ywidth='+ST+';'+#13#10);
33:    Xwidth:=Xright-Xleft+1;
34:    Str(Xwidth,ST); SaveData('      Xwidth='+ST+';'+#13#10);
35:    SaveData(' Crut:array[1..Ywidth,1..Xwidth*3]'+
36:                                ' of byte='#13#10'   (');
37:    Y:=Yup;
38:  repeat
39:     No:=1;
40:    for x:=Xleft to XRight do
41:      begin
42:           Y:=Yup;
43:         for round:=0 to 2 do
44:           begin
45:            CrutData:=0; setcolor(Round+11);
46:            for k:=0 to 7 do
47:             if getpixel(x,y+k)<>0 then
48:                   CrutData:=CrutData+Bitk[k];
49:               line(x,y,x,y+7);
50:               str(crutdata:3,ST);
51:               if ST[1]=#32 then ST[1]:='0';
52:               if ST[2]=#32 then ST[2]:='0';
53:               ST:=ST+',';
54:               if no=1 then ST:='('+ST;
55:               if (no mod 11=0) and (No<Xwidth*3) then ST:=ST+#13#10'     ';
56:              SaveData(ST); inc(no);
57:            inc(Y,8);
```

รูปที่ 6 (ต่อ)

```
58:           end;
59:      end;
60:       Yup:=y;  seek(f,filesize(f)-1);
61:       if y<Ydown then Savedata('),'#13#10'     ')
62:                  else  Savedata('));'#13#10#26);
63:  until Y>Ydown;
64:   close(f);
65: end;

66: {============== TRANCRUT.PAS ==============}
67: begin
68:  ReadCrut;
69:  trandata;
70:  closegraph;
71: end.
```

เมื่อหารด้วย 24 จะลงตัวพอดี หากค่านี้ไม่ลงตัวจะมีปัญหาในการเก็บไม่ถูกต้อง

4. บรรทัดที่ 15 กำหนดค่าคงที่เพื่อใช้ตรวจสอบการลงจุดด้วยคำสั่ง getpixel

5. บรรทัดที่ 25-26 เปิดไฟล์ชื่อ crutdata.pas เพื่อเก็บข้อมูลภาพเพื่อนำไปพิมพ์

6. บรรทัดที่ 27 กำหนดค่ามุมบนซ้ายของภาพที่จะเก็บ

7. บรรทัดที่ 28 กำหนดค่ามุมล่างขวาของภาพที่จะเก็บ

8. บรรทัดที่ 29-30 ตรวจขอบเขตของค่า Yup และ Ydown เพื่อให้เป็นไปตามที่กำหนด

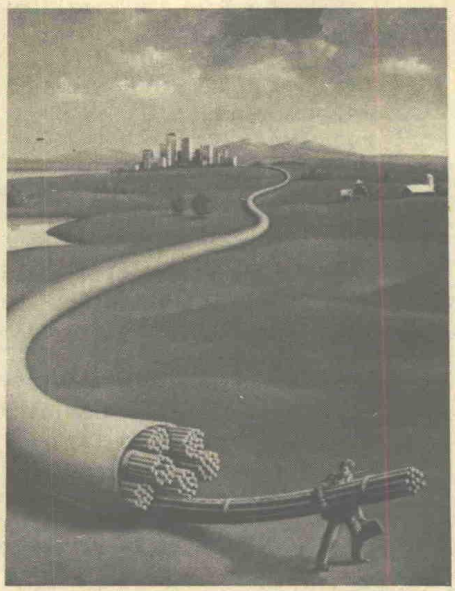
9. บรรทัดที่ 31-32 หาจำนวนแถวของการพิมพ์พร้อมกับเขียนลงไฟล์ crutdata.pas เพื่อกำหนดค่าคงที่ของ Ywidth

10. บรรทัดที่ 33-34 หาความกว้างของการพิมพ์แต่ละแถวพร้อมกับเขียนลงไฟล์

11. บรรทัดที่ 35-36 เขียนขอบเขตมิติของค่าคงที่ crut ลงไฟล์ ซึ่งคราวนี้จะไม่เป็นตัวเลข แต่เป็นการกำหนดขอบเขตมิติด้วยตัวแปร การที่กำหนดให้เป็น Xwidth*3 เพราะว่าในการสั่งพิมพ์ของเครื่องพิมพ์ที่มี 24 หัวเข็มนั้นหนึ่งครั้งในการพิมพ์ใช้ข้อมูล 3 ไบต์ ดังนั้นในหนึ่งแถวข้อมูลจึงต้องมีค่าเป็น 3 เท่าของความกว้างที่คำนวณได้

12. บรรทัดที่ 38-63 วนรอบโอนข้อมูลเก็บเข้าไฟล์

13. บรรทัดที่ 40-59 การเก็บข้อมูลการพิมพ์หนึ่งแถว ซึ่งเป็นการเก็บภาพครั้งละ 24 จุด (แนวดิ่ง)



14. บรรทัดที่ 43-58 วนรอบการเก็บข้อมูลครั้งละ 8 จุด รวม 3 ครั้ง นับจากบนลงล่างจึงทำให้เก็บได้ 24 จุด โดยข้อมูลที่เก็บลงไฟล์แล้วจะถูกเปลี่ยนมาเป็นเส้นตรงตามคำสั่งบรรทัดที่ 49

15. บรรทัดที่ 50-53 เปลี่ยนข้อมูลที่หาได้เป็นข้อความในตัวแปร ST

16. บรรทัดที่ 54 กรณีที่เป็นข้อมูลแรกของแถวการพิมพ์ให้เพิ่ม (ที่ต้นข้อความ

17. บรรทัดที่ 55 ให้เลื่อนบรรทัดทุก ๆ 11 ข้อมูล (ยกเว้นกรณีที่ no = Xwidth*3)

18. บรรทัดที่ 56-57 เก็บข้อมูลลงไฟล์ crutdata.pas และเลื่อนค่าตัวแปร y เพื่อหาข้อมูลลำดับถัดไป

19. บรรทัดที่ 60 เลื่อนพอยน์เตอร์การเขียนไฟล์ถอยหลังมาหนึ่งไบต์ เหตุผลที่เลื่อนก็เพราะว่าขณะนี้โปรแกรมมาถึงข้อมูลสุดท้ายของแถวการพิมพ์แล้ว แต่ในไฟล์ได้ถูกเขียนเครื่องหมาย , ไว้ที่ท้ายสุดของไฟล์ซึ่งไม่ถูกต้อง เพราะต้องลงท้ายด้วยวงเล็บปิด) เท่านั้น เราจึงเลื่อนตำแหน่งเขียนไฟล์ขึ้นมาหนึ่งระเบียนข้อมูล

20. บรรทัดที่ 61 เขียนส่วนปิดท้ายแถวของข้อมูลและเลื่อนบรรทัด

21. บรรทัดที่ 62 เขียนส่วนปิดท้ายข้อมูลและจบไฟล์

22. บรรทัดที่ 76-71 ส่วนการเรียกโปรแกรมที่เขียนใช้งาน

ลักษณะของข้อมูลที่เก็บในไฟล์ crutdata.pas เมื่อรันไฟล์ trancrut.pas สิ้นสุดลงนั้นจะเป็นดังตัวอย่างในรูปที่ 7

ผู้อ่านอาจลองทำการโอนภาพครุฑอีกขนาดหนึ่ง (เล็กกว่า) จากไฟล์ crut2.pic ดูก็ได้ โดยให้เขียนรูปสี่เหลี่ยม

```
Const Ywidth=9;
      Xwidth=201;
 Crut:array[1..Ywidth,1..Xwidth*3] of byte=
   ((000,000,000,000,000,000,000,000,000,000,000,
     001,000,000,015,000,000,255,000,001,252,000,
                          .
                          .
     000,015,000,000,001,000,000,000,000,000,000,
     000,000,000,000,000,000,000,000,000),
                          .
                          .
     000,000,000,000,000,000,000,000,000,000,000,
     000,000,000,000,000,000,000,000,000));
```

รูปที่ 7 ส่วนหนึ่งของข้อมูลที่โอนได้จากรูปครุฑบนจอภาพ

3. บรรทัดที่ 5-13 ชุดคำสั่งส่งข้อมูลที่อ่านได้ไปพิมพ์

4. บรรทัดที่ 14-25 ชุดคำสั่งในการพิมพ์ข้อมูลจากไฟล์ crutdata.pas ลงเครื่องพิมพ์

5. บรรทัดที่ 17 กำหนดระยะกั้นซ้ายในการพิมพ์

6. บรรทัดที่ 18 คำนวณจำนวนข้อมูลในการพิมพ์หนึ่งแถวจากค่าคงที่ Xwidth

7. บรรทัดที่ 19-24 ส่วนการพิมพ์ข้อมูลทั้งหมดจำนวน Ywidth ครั้ง

8. บรรทัดที่ 21 กำหนดระยะบรรทัดและเลือกพิมพ์กราฟิกโหมด 39 ขนาด 180 จุดต่อนิ้ว ซึ่งทำให้ได้ผลการพิมพ์ตรงกับความจริงของภาพที่โปร่งดูขอบเขตของภาพเสียก่อน หรือจะลองโอนภาพอื่น ๆ ที่ผู้อ่านมีอยู่ก็ได้ เพื่อจะได้เข้าใจหลักการเก็บข้อมูลได้ดีขึ้น แล้วท่านจะเห็นว่าโปรแกรมสั้น ๆ นี้มีประโยชน์มหาศาลทีเดียว

สร้างไฟล์พิมพ์รูปครุฑ

โอนข้อมูลรูปครุฑจากจอภาพเก็บลงไฟล์ crutdata.pas เรียบร้อยแล้ว ถ้าเราไม่เขียนโปรแกรมนำข้อมูลนั้นมาใช้งานก็จะเป็นการเสียเวลาไปเปล่า ๆ ดังนั้นผู้เขียนจึงขอจบลงด้วยการนำข้อมูลที่เก็บไว้นี้มาพิมพ์ลงเครื่องพิมพ์ 24 หัวเข็มต่อไป โดยจะเขียนสั้น ๆ ดังโปรแกรมในรูปที่ 8 มีรายละเอียดดังนี้

1. บรรทัดที่ 3 เรียกไฟล์ crutdata.pas มาคอมพายล์ร่วมด้วยคำสั่ง {$I crutdata}

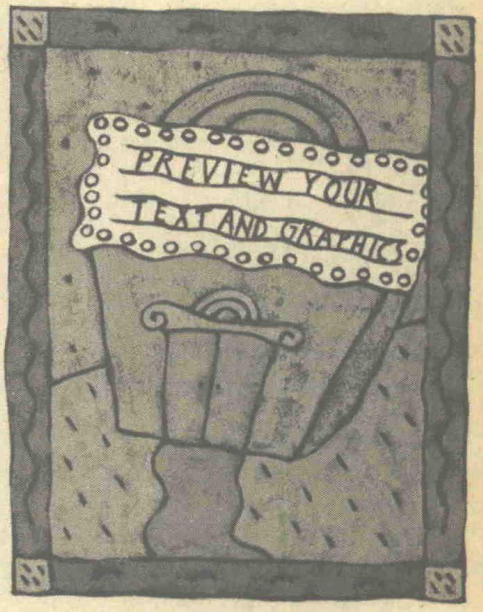
2. บรรทัดที่ 4 กำหนดตัวแปรกั้นซ้ายในการพิมพ์

รูปที่ 8 โปรแกรมพิมพ์รูปครุฑจากข้อมูลที่โอนได้ (ไฟล์ crutdata.pas)

```
 1: PROGRAM Print_Crut;
 2: uses crt,dos;
 3: {$I CrutData}
 4: var LeftMargin:integer;

 5: PROCEDURE Print(DataST:string);
 6: var  Regs:Registers; B:byte;
 7: begin
 8:   for B:=1 to length(DataST) do
 9:     begin
10:       Regs.AH:=0;       Regs.AL:=ord(DataST[b]);
11:       Regs.DX:=Pred(1); Intr($17, Regs);
12:     end;
13: end;

14: PROCEDURE PrintCrut;
15: var y,n1,n2,no:word;
16: begin
17:    Print(#27#80#27'l'+chr(leftmargin));
18:    n1 := Lo(Xwidth);  n2 :=Hi(Xwidth);
19:  for y:=1 to Ywidth do
20:   begin
21:     Print(#27'3'#24#27'*'+#39+chr(n1)+chr(n2));
22:     for no:=1 to Xwidth*3 do  Print(chr(Crut[y][no]));
23:     Print(#10);
24:   end;
25: end;

26: {==================== CRUT.PAS ======================}
```

รูปที่ 8 (ต่อ)

```
27: begin
28:  if Paramstr(1)='' then
29:   begin
30:     clrscr;
31:     writeln('========== PRINT CRUT ==========');
32:     writeln('     Please turn on Printer ');
33:     repeat
34:       write('Let Leftmargin (4..55) ');
35:       Read(Leftmargin);
36:    until Leftmargin in [4..55];
37:   end
38:    else begin
39:           val(Paramstr(1),leftmargin,leftmargin);
40:           if leftmargin=0 then leftmargin:=4;
41:         end;
42:   PrintCrut;
43: end.
```



รูปที่ 9
รูปครุฑและภาพอื่น ๆ
ที่พิมพ์ได้จาก
แนวคิดของ
บทความนี้

เราสแกนมาพอดี

9. บรรทัดที่ 22 ส่งข้อมูลทั้งแถวไปพิมพ์ลงกระดาษ

10. บรรทัดที่ 23 เมื่อพิมพ์จบหนึ่งแถวก็ทำการเลื่อนบรรทัด

11. บรรทัดที่ 27-43 ส่วนโปรแกรมหลักในการทำงาน ซึ่งจะทำได้ 2 ลักษณะคือ ถ้ารันไฟล์โดยป้อนเพียงชื่อไฟล์ จะให้ป้อนระยะกั้นซ้ายบนจอภาพ ถ้ารันไฟล์พร้อมให้ระยะกั้นซ้ายด้วยก็ได้

เมื่อคอมพายล์ไฟล์นี้แล้ว จะได้ไฟล์ crut.exe ที่มีข้อมูลการพิมพ์อยู่ในไฟล์แล้ว จึงสามารถสำเนาไปใช้งานในแผ่นดิสค์อื่น ๆ ได้เลย

จากการกำหนดให้ป้อนพารามิเตอร์ต่อท้ายไฟล์ได้ จึงสามารถนำไฟล์นี้ไปเรียกใช้ผ่านโปรแกรมพิมพ์เอกสารราชวิถีเวิร์ดพีซีได้ โดยการใช้ป้อน .!crut 30 ในแฟ้มเอกสาร นั่นคือผู้อ่านสามารถนำไปสร้างโลโกหัวจดหมายโลโกสถาบันหรือหน่วยงานของท่านได้

เพื่อให้เห็นผลการทำงาน ผู้เขียนจึงขอนำภาพที่พิมพ์ได้จากแนวคิดนี้มาให้ดูเป็นตัวอย่างดังรูปที่ 9 ซึ่งรูปครุฑทั้งสองก็คือการพิมพ์จาก crut1.pic, crut2.pic ส่วนรูปอื่น ๆ นั้นเป็นโลโกของโรงเรียนต่าง ๆ ที่เคยทำให้

ก่อนจาก

หวังว่าต่อไปนี้ ผู้อ่านคงได้ปรับปรุงโปรแกรมเก่าที่เขียนในลักษณะอ่านข้อมูลข้างนอก มาเป็นการนำข้อมูลนั้น ๆ มาผนวกไว้ในไฟล์ EXE ของท่านต่อไป

พบกันคราวหน้าด้วยเรื่องที่น่าสนใจเช่นเดิม